Software Development Products

Software Defect Update

Intel® C++ Compiler for Linux* and Windows*

2nd March 2005

Legal Disclaimer

If you have any questions about the issues discussed in this report, please open a support issue at https://Premier.intel.com.

Linux*

21517	Template function with too few arguments is accepted
21708	Incorrect read, write and execute permissions set on compiler files
22064	FlexLM* cannot find license file if directory path contains @
23916	Double data types are 80 bits rather than 64 bits – no way to get 64–bit double data types
46823	Failure to call terminate() for an exception that is thrown from a copy constructor
24094	Problem with object references in constructor
25004	Loop with a switch statement is not auto–parallelized
25005	Loop with a conditional operator(?:) is not parallelized
25430	C++ compiler incompatible with SGI* hash_set, hash_map implementation
25416	Compiler reports incorrect error message when compiling with the invalid switch –Xk
26085	The icc compiler generates shared object files that are 3 to 6 times larger than gcc on STL code.
26654	Configuration scripts fail to setup man path; man –w does not work
26302	Inlining is disabled when the address of one of the function formal parameters is referenced
26379	Compiler may not inline all mathematical functions
26702	Cannot do multi–file IPO when using a library.
26589	Incorrect DWARF debug information generated for C99 adjustable arrays
26740	No error message is displayed when you mistakenly pass wrong argument type to the atoi function
26994	Profile Guide Optimization ineffectual on files containing inline assembly
27633	libraries built using the –g swtich are enormous
29341	The Itanium(R) compiler hangs compiling lame MP3 encoder at –O2 optimization level
28049	Scope of types wrong in C++ debug information
28784	Cannot make a symbol "weak"
29495	Compiler reports error : "non–integral operation not allowed in nontype template argument"
29794	icc reports segmentation violation when compiling code containing unnamed structures
30326	Support for non–standard anonymous unions
31180	The port@host implementation for floating license servers may not work

31500	"undefined reference" error when using kmp_set_stacksize_s macro in omp.h header file
29998	Code using cross cast causes Segmentation fault at run time
31640	'–help' option incorrectly states default internal floating–point precision
31792	Internal Error when declaring static string objects in a static function and compiling with icc –g
32058	The –fno–gnu–keywords option is not supported
32170	Compiler doesn't support assignment of std::auto_ptr operand types
32179	Compiler produces incorrect code when using the dynamic_cast operator
28771	Compiler reports errors if no file names are specified
32285	Scope of variables in for loops
32373	Compiler cannot differentiate typedefs and function parameters with the same names
32405	The Compiler for Itanium(R)–based Applications doesn't Implement –nolib_inline
32883	The exception::what() member function rerturns incorrect result
33664	Compiler generates remark #810 for computed if with bitfield
34012	OpenMP code fails to link if 'write' is used as name for critical region
34565	Compiler reports segmentation violation when compiling C++ files at –ip –ipo
38180	# and * in asm constraint modifier is not accepted
43784	Internal compiler error while compiling CERN*'s ROOT* application version 3.10.2 at –O3 –ipo
37717	INTEL (flexlm) vendor daemon fails to start
43958	Premature program termination when using shared memory segments greater than 2GB
48406	Incorrect results produced when code containing complex arithmetic is optimized
48869	__interface variable in some new versions of glibc conflicts with compiler internal

Windows*

18282	Derived class enumeration bug
19189	No option to turn off "targeted for automatic cpu dispatch" remarks.
20109	printf invalid format string: warning #269
21906	Itanium(R) compiler unaligned access error with __unaligned keyword used on a structure

22666 Compiler does not support double casting

22910 Dependency information not generated under certain circumstances

23572 Large value (>=0x80000000) cast from double to int produces incorrect results

25496 Apps with Deep Class Inheritance such as STLport will take Longer to Compile

24227 The C++ Compiler compiles an illegal member–like constructor call

25485 Compiler issues error message when compiling in–line asm code using __asm_emit

25649 Itanium(R) compiler treats the labs() intrinsic function like a regular function call

26318 Runtime errors when using Stingray* Objective Studio 2000 with the C++ Compiler for Windows*

28272 icl ignores the Using–declaration of base class assignment operator

29485 Pragma warning directive turns all warnings to errors

30900 Definition for explicit instantiations of template functions is required

29945 No warning message issued when 'delete' is called on a local array

41075 Preprocessing goes into endless loop when the source includes itself

31326 Reference to zero–sized array not reported as error

31487 Additional parentheses in sizeof() argument causes the Intel(R) C++ Compiler to diagnose an error

28538 extern "C" comment() definition compiled with –Zi causes an internal compiler error

21753 Unused static functions are not removed and may cause link errors

33117 Aligned and unaligned intrinsics not differentiated when inlined

32992 The _OPENMP predefined macro is not documented

33423 PGO Performance Counter Overflow Causes Erroneous "Low Trip Count" Diagnostics

33931 7.1 IDE Integration Overwrites 6.0 Integration

34091 Pure Virtual Functions Cannot Be Set With Integer Specifiers

34271 dllexport Attribute not Correctly Applied to Explicit Template Specializations

16886 Compile error when both a template copy constructor and a template conversion "operator =" defined

34653 std::allocator::rebind compilation error

33872 Unresolved Symbols with –MD[d] and Microsoft* Visual C++* .NET 2003

34674	Intel Specific property does not show under Microsoft* Visual Studio .NET 2002 Japanese Edition
36069	/GL (Whole Program Optimization) switch may increase object size
37449	Internal Compiler Error on Code with Multiple goto Statements
37176	Unresolved symbol error "__VEC_memzero" when calling the memset function
38477	The FFTW library does not compile when using the –Qipo switch
38225	Problem with project rebuilding when ici80.dg is deleted
38704	The _mm_malloc() intrinsic function doesn't return zero on failure
41535	The Itanium(R) Assembler (ias) does not flag error for unbalanced curly or square brackets
39854	Unresolved external symbol ___readfsdword referenced
45537	Abstract virtual function called from constructor not diagnosed until runtime
45977	Name–mangling of __stdcall Constructors/Destructors with Mixed Builds
48977	Error LNK2001 when compiling projects created by ATL Wizard
49660	Missing copy constructor instantiation at –Od when the class contains "__declspec(dllimport)"
49807	Wrong vectorization if "#pragma novector" is not used in test case

Linux*

Title	Template function with too few arguments is accepted
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.2

Problem Description	The following program compiles without any errors, although the function "wibble" can never be called with only one argument. Also, the default arguments from the function "bar" are taken over into the call of "wibble", which is incorrect.
	
	template<typename F>
	void foo(F f)
	{
	f(1);
	}
	
	void bar(int i, int j = 5, int k = 7)
	{

	std::cout << "bar(" << i << ", " << j << ", " << k << ")" << std::endl;
	}
	
	void wibble(int i, int j, int k = 9)
	{
	std::cout << "wibble(" << i << ", " << j << ", " << k << ")" << std::endl;
	}
	
	int main()
	{
	foo(
	foo(// should give an error, "wibble" cannot be called with one arg
	}

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Incorrect read, write and execute permissions set on compiler files
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	The installation sets incorrect file permissions on the compiler files. For example the compiler documents under the docs directory all have execute file permissions set.
Resolution/Status	This is resolved in the 8.0 compilers.

Title	FlexLM* cannot find license file if directory path contains @
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	If the Intel(R) C++ Compiler for Linux* is installed into a directory path that contains '@' (for example, /opt/@cell/compiler), FlexLM* cannot find the license file.
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround remove the '@' symbol from the directory path.

Title	Double data types are 80 bits rather than 64 bits – no way to get 64–bit double data types
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1

Problem Description	The switch –long_double is on by default, enabling 80–bit long double type. There is no way to disable this switch and get 64–bit double type.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Failure to call terminate() for an exception that is thrown from a copy constructor
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	The Intel(R) C++ Compiler for Itanium(R)–based applications (ecc) does not call terminate() while handling an exception that is thrown from a copy constructor.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Problem with object references in constructor
Product	Intel(R) C++ Compiler for Linux*
Version	5,0,6.0,5.0,6.0
Operating System	Red Hat* 7.1
Problem Description	The methods and data members of an object should be accessible from the constructor of an object. ecc produces SEGV and incorrect results for various references to object methods and data members in the object constructor.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Loop with a switch statement is not auto–parallelized
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1

Problem Description	A loop containing a switch statement is not auto–parallelized, when compiling with the –parallel option.
	An example is provided below.
	
	main() {
	int i;
	for (i=0; i<50; i++) {
	switch(i){}
	}
	
	}

	icc –parallel –par_report3 test.c test.c procedure: main serial loop: line 3: not a parallel candidate due to the loop being lexically discontinuous
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Loop with a conditional operator(?:) is not parallelized
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1

Problem Description	If conditional operator (?:) appears in a loop, the loop is not auto–parallelized.
	
	#include <stdio.h>
	main() {
	int j;
	int a[100];
	for (j=0; j<50; j++) {
	a[j]=(j>25)? 1 : 0;
	}
	printf("%d\n",a[44]);
	
	}
	
	icc –parallel –par_report2 test.c
	test.c
	procedure: main
	serial loop: line 5: not a parallel candidate due to unknown reasons
	serial loop: line 5
	output data dependence assumed from line 6 to line 6
	output data dependence assumed from line 6 to line 6
	flow data dependence assumed from line 6 to line 6
	output data dependence assumed from line 6 to line 6
	output data dependence assumed from line 6 to line 6
	flow data dependence assumed from line 6 to line 6
	anti data dependence assumed from line 6 to line 6
	anti data dependence assumed from line 6 to line 6

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	C++ compiler incompatible with SGI* hash_set, hash_map implementation

Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 6.2
Problem Description	Code written to use the hash_map and hash_set interfaces provided by GNU* and SGI* C++ compilers does not compile with the Intel C++ compiler.
Resolution/Status	The hash_set and hash_map classes are not part of the C++ standard library. Therefore, there are multiple conflicting implementations. The Intel(R) C++ Compiler uses the Dinkumware* implementation. To compile under the Intel C++ compiler, use the interface provided by Dinkumware*. This interface is described in documentation available at their web site at http://www.dinkumware.com/htm_cpl/hash_map.html.

Title	Compiler reports incorrect error message when compiling with the invalid switch –Xk
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	If you use the invalid compiler switch "–Xk" to compile a simple C program containing a reference to the stdio.h header file, the compiler issues the following invalid error messages: icc –c –Xk ers.c catastrophic error: #error directive: "You need a ISO C conforming compiler to use the glibc headers" # error "You need a ISO C conforming compiler to use the glibc headers" error: expected a ";" typedef signed char __int8_t; error: invalid combination of type specifiers __extension__ typedef signed long long int __int16_t; error: invalid combination of type specifiers __extension__ typedef signed long long int __int32_t; ^ error: invalid combination of type specifiers __extension__ typedef signed long long int __int64_t; "/usr/include/gconv.h", line 71: error: identifier "const" is undefined __const unsigned char **, __const unsigned char *,

	^
Resolution/Status	This is a known issue that may be resolved in a future product release. The correct compiler switch to use is –xK (lower case x and capital K).

Title	The icc compiler generates shared object files that are 3 to 6 times larger than gcc on STL code.
Product	Intel(R) C++ Compiler for Linux*
Version	P6.0
Operating System	Red Hat* 8.0
Problem Description	When code that make heavy use of STL is compiled to produce a shared object, the code size of the shared object is three to six times bigger than that produced by gcc.
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround do not use shared objects and use static linking.

Title	Configuration scripts fail to setup man path; man –w does not work
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	SGI Altix ProPack 3.0
Problem Description	The MANPATH variable is required to display the man page for the Intel Compiler. If MANPATH is not defined when the configuration script executes, the script uses the "man –w" command to acquire the location of the man files. Some versions of the man function do not recognize the "–w" switch, ignoring the switch and displaying the prompt "What manual page do you want?". This causes the script to terminate prematurely, without setting the MANPATH variable. This is known to be a problem with man 2.3.19, which has come with SuSE* Linux* 7.3.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Inlining is disabled when the address of one of the function formal parameters is referenced
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	The Intel(R) C++ Compiler for Itanium(R)–based applications (ecc) disables function inlining when the address of one the function formal parameters is assigned to a local variable.

	For example: inline void tobuf(char *UShort_t x) { char *sw = (char *)x <=== address of x. The Intel(R) C++ Compiler for 32−bit applications, does not have this limitation.
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_bc_8.0.047 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit

Title	Compiler may not inline all mathematical functions
Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1

Problem Description	The Intel(R) C++ Compiler for Itanium(R)−based applications does not inline math functions, apart from sqrt, and consequently does not utilize the parallel instruction scheduling capability of the Itanium(R) processor. This leads to a slow down in these kinds of math−intensive functions.
	
	#include<math.h>
	#include<iostream.h>
	void main()
	{
	double f=25, g;
	g = sin(f); //this call is not inlined
	cout<}

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.055 or higher. You may download and install the latest product update from the Premier Support web site at . You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Cannot do multi−file IPO when using a library.
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat* 7.1
Problem Description	If code is compiled separately (with −ipo) and placed in a library, the compiler will not perform any multi−file optimization when a program links against the

	library (and is itself also compiled with –ipo). Instead, everything degrades to single–file optimization.
	
	The following example illustrates this problem and shows the diagnostic message output by the compiler:
	
	Compile main.cpp by itself with –ipo, use xild to put the resulting object in a static library, and then link with ecc and only single–file optimizations are indicated:
	
	$ ecc –c –ipo main.cpp
	main.cpp
	
	$ xild –lib cru mylib.a main.o
	IPO: using IR for main.o
	IPO: performing single–file optimizations
	xiar: executing 'ar'
	
	$ ecc –ipo shrtvect.cpp mylib.a
	IPO: using IR for shrtvect.o
	IPO: performing single–file optimizations

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.055 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit

Title	Incorrect DWARF debug information generated for C99 adjustable arrays
Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1
Problem Description	The compiler generates incorrect DWARF debug information for C99 adjustable arrays. Specifically: – It describes all of the array objects as references to adjustable arrays, rather than as arrays themselves; – all of the variables which are dynamic arrays are described in the DWARF as arguments to the subroutine in which they occur, even if they're declared as locals.
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pu_7.0.082 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information,

	please visit http://www.intel.com/software/products/support

Title	No error message is displayed when you mistakenly pass wrong argument type to the atoi function
Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1

Problem Description	The compiler does not display any error messages when you pass a 'char' argument to the atoi function by mistake. Instead, the compiler issues a warning message and the program may crash at run−time. The GNU* compilers, however, report an error message and do not compile the program as shown below:
	
	int main()
	{
	char c = 'x';
	int i = atoi(c);
	return 0;
	}
	
	icc test_atoi.cpp
	test_atoi.cpp:6: cannot convert `char' to `const char*' for argument `1' to `int atoi(const char*)'

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Profile Guide Optimization ineffectual on files containing inline assembly
Product	Intel(R) C++ Compiler for Linux*
Version	6.0, 7.0
Operating System	Red Hat 7.2
Problem Description	Compiling a module containing inline assembly with profile−guided optimization (PGO) options is ineffectual. The resulting object file shows no benefit from the PGO options.
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.055 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support.

Title	libraries built using the −g swtich are enormous

Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1
Problem Description	Building libraries with –g result in enormous file size of over 10x versus not using –g.
Resolution/Status	This is a known issue and may be resolved in a future product release.

Title	The Itanium(R) compiler hangs compiling lame MP3 encoder at –O2 optimization level
Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1
Problem Description	The Itanium(R) compiler hangs when compiling the "lame MP3 encoder Linux* application" at the –O2 optimization level, but compiles correctly at –O0, –O1, and –O3. The problem occurs during the compilation of the decode_i386.c module.
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pc_8.0.058_pl059 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Scope of types wrong in C++ debug information
Product	Intel(R) C++ Compiler for Linux*
Version	6.0
Operating System	Red Hat* 7.1

Problem Description	The compiler emits type definitions in the wrong scope in the DWARF debug information.
	
	struct outer {
	struct inner {
	struct innerinner {
	int x;
	} inner_innerinner;
	} outer_inner;
	};
	
	In this example, the scope of the definition of struct inner is struct outer, not global scope as reported by the compiler.

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.055 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support.

Title	Cannot make a symbol "weak"
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1
Problem Description	The Intel(R) C++ Compiler for Linux* documentation states that the compiler does not support "Function Attributes Declarations" such as the use of the "weak" attribute shown below: extern void foo(void) __attribute__((weak));
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.055 or higher. You may download and install the latest product update from the Premier Support web site at . You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Compiler reports error : "non−integral operation not allowed in nontype template argument"
Product	Intel(R) C++ Compiler for Linux*
Version	8.0
Operating System	Red Hat* 7.1
Problem Description	The Intel(R) C++ Compiler for Linux, Version 8.0, does not allow non−integral operation in nontype template argument. The following code illustrates the encountered problem. class A {}; class B : A {}; template<const A* a> class C {}; template<const B* b> class D {}; template<B* b> class E {}; template<const B* b> void f(D **,C(b)>){}** **^** **When compiled, the compiler generates the error message** **error: non−integral operation not allowed in nontype template argument**

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	icc reports segmentation violation when compiling code containing unnamed structures
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1
Problem Description	The compiler aborts program compilation and reports a segmentation violation error when compiling code that contains unnamed structures as shown in the following example: test.cpp −−−−−−−−−−−− typedef struct { struct s; } abc; icc –c test.cpp icc: error: Fatal error in /opt/intel/compiler70/ia32/bin/mcpcom, terminated by segmentation violation compilation aborted for test.cpp (code 1)
Resolution/Status	This problem has been resolved in the Intel C++ compiler version 8.0. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit

Title	Support for non−standard anonymous unions
Product	Intel(R) C++ Compiler for Linux*
Version	6.0,7.0
Operating System	Red Hat* 7.1

Problem Description	The following code gives an error: union "Vector3" has no member "x0"
	
	template <typename T>
	class Vector3
	{
	union {
	struct {

	T x0;
	T x1;
	T x2;
	};
	T r[3];
	} _arr;
	Vector3()
	{
	_arr.x0=0;
	}
	};
	
	Vector3<int> intvec;
	
	This error is due to the fact that the Intel(R) C++ Compiler does not support non−standard anonymous unions.

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pu_7.0.082 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/

Title	The port@host implementation for floating license servers may not work
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.2 − delete
Problem Description	When configuring a floating license server, if the client license is configured such that INTEL_LICENSE_FILE = port@host to find the license server, the compiler may not recognize the license file as valid, displaying an error message such as: Error: A license for CComp is not available (−18,147).
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, place the actual license file on the client system and link the INTEL_LICENSE_FILE variable directly to that file. You may also place the file on an NFS mounted share, and point all the clients to that same file (to save on file replication).

Title	"undefined reference" error when using kmp_set_stacksize_s macro in omp.h header file
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1

Problem Description	"undefined reference" error is reported when compiling code that references the macro kmp_set_stacksize_s defined in the omp.h header file: #include <omp.h> #include <stdlib.h> #include <stdio.h> main() { long stacksize; stacksize = atol(getenv("KMP_STACKSIZE")); if (stacksize < 10000000) kmp_set_stacksize_s(10000000); printf("Stacksize = %lld\n", kmp_get_stacksize_s()); } % ecc –openmp test.c /tmp/ecclNUBt5.o: In function `main': /tmp/ecclNUBt5.o(.text+0x142): undefined reference to `kmpc_set_stacksize_s'
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_7.1.006 or higher. You may download and install the latest product update from the Premier Support web site at . You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Code using cross cast causes Segmentation fault at run time
Product	Intel(R) C++ Compiler for Linux*
Version	7.0, 7.1
Operating System	Red Hat* 7.1

Problem Description	Code that uses cross cast (casting type S to type T with T being a sibling of S in a multiple inheritance hierarchy) fails at run time with a Segmentation fault:
	
	// The following code compiles, but fails at run time
	#include <iostream>
	#include <assert.h>
	using std::cout;
	
	struct B1 {
	virtual char kind1() { return '1'; }

	};
	
	struct B2 {
	virtual char kind2() { return '2'; }
	};
	
	struct Derived : public B1, public B2 // NOTE: order is B1,B2
	{
	virtual char kind() { return 'D'; }
	};
	
	int main()
	{
	B1 * b1 = new Derived;
	B2 * b2 = NULL;
	
	b2 = dynamic_cast<B2*>(b1); // crosscast
	assert(b2–>kind2() == '2');
	return 0;
	}
	
	$ icc testfail.cpp
	$ ./a.out
	Segmentation fault

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pu_7.1.011 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support .

Title	'–help' option incorrectly states default internal floating–point precision
Product	Intel(R) C++ Compiler for Linux*
Version	7.0, 7.1
Operating System	Red Hat* 7.1
Problem Description	The compiler command line help (e.g. icc –help) incorrectly notes that the default internal FPU precision is –pc64 (53 bit significand). The default is actually –pc80 (64 bit significand).
Resolution/Status	This problem has been resolved in the Intel(R) C++ Compiler 7.1 l_cc_pc_7.1.017 or later. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need

	to be a registered user to access Premier Support. For registration information, please visit http:/www.intel.com/software/products/support.

Title	Internal Error when declaring static string objects in a static function and compiling with icc –g
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1

Problem Description	When you compile the following sample code using the –g switch, the compiler reports an internal compiler error as shown below:
	
	#include <string>
	
	static void test()
	{
	static std::string r;
	}
	
	icc –g test.cpp
	/opt/intel/compiler70/ia32/include/xlocinfo(59): (col. 2)internal error: assertion failed at: "proton/edgglue/edg_main.c", line 593
	
	compilation aborted for test.cpp (code 4)

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pu_7.1.013 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	The –fno–gnu–keywords option is not supported
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.3
Problem Description	A test case compiles with GNU g++ using "g++ –fno–gnu–keywords –o hello hello.cpp". The testcase does not compile with the Intel(R) C++ Compiler because the test case uses 'typeof' as keyword. The Intel C++ Compiler does not have an equivalent option to –fno–gnu–keywords option in gcc.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Compiler doesn't support assignment of std::auto_ptr operand types
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	Red Hat* 7.1

Problem Description	The compiler reports errors when processing code that uses the operand type std::auto_ptr<int> as in the following example:
	
	#include <stdio.h>
	#include <iostream>
	#include <memory>
	
	main()
	{
	std::auto_ptr<int> b(std::auto_ptr(new int));
	std::auto_ptr<int> c;
	c = b;
	b = std::auto_ptr<int>(new int);
	}
	
	$ icc –o test70 test.c
	
	test.c(11): error: no operator "=" matches these operands
	operand types are: std::auto_ptr<int> = std::auto_ptr
	b = std::auto_ptr<int>(new int);
	^

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pu_7.1.011 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support .

Title	Compiler produces incorrect code when using the dynamic_cast operator
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* AW 2.1
Problem Description	The compiler produces incorrect code when compiling a program that uses the dynamic_cast operator and multiple inheritance as shown below: #include <iostream> class A { public: virtual void foo() {}

	};
	
	class B
	{
	public:
	virtual void bar() {}
	};
	
	class Derived : public A, public B
	{
	};
	
	int main(int argc, char *argv[])
	{
	Derived* d = new Derived;
	A* a = d;
	if (dynamic_cast<B *> (d))
	std::cout << "Object " << d << " is derived from B. "
	<< "Ecc will print this." << std::endl;
	
	if (dynamic_cast<B *> (a))
	std::cout << "Object " << a << " is derived from B. "
	<< "Ecc will NOT print this!" << std::endl;
	
	delete d;
	
	return 0;
	}
	
	$ ecc –O0 eccBug.cpp
	$ ./a.out
	Object 6000000000017eb0 is derived from B. Ecc will print this.
	
	When compiled correctly, the correct result should be:
	
	Object 0x6000000000001fa0 is derived from B. Ecc will print this
	Object 0x6000000000001fa0 is derived from B. Ecc will NOT print this!

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_7.1.009 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Compiler reports errors if no file names are specified
Product	Intel(R) C++ Compiler for Linux*
Version	7.0, 7.1
Operating System	Red Hat* 7.1
Problem Description	The compiler reports the following message when no file names are provided as arguments: $ icc /usr/lib/crt1.o: In function `_start': /usr/lib/crt1.o(.text+0x18): undefined reference to `main'
Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pc_8.0.066 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Scope of variables in for loops
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	Red Hat* 7.3

Problem Description	The attached test case fails to compile with icpc but compiles with g++ using the –fno–for–scope command line option.
	
	––
	// test.cpp
	
	#include <stdio.h>
	
	int main()
	{
	for(int i= 0; i < 5;){
	++i;
	}
	printf("hello world. = %d\n", i);
	return 0;
	}
	––

Resolution/Status	This is resolved in the 8.0 compilers.

Title	Compiler cannot differentiate typedefs and function parameters with the same names
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	Red Hat* 7.3
Problem Description	If a function parameter name has been previously used in a typedef declaration, the compiler reports an error as shown in the following example: typedef int itype; typedef void (*f) (int itype, itype i); $ icc –c test.c test.c(2): error: parameter "itype" is not a type name typedef void (*f) (int itype, itype i); ^ compilation aborted for test.c (code 2)
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, do not use the same typedef name for a function parameter. For example: typedef int itype; typedef void (*f) (int itype_new, itype i);

Title	The Compiler for Itanium(R)–based Applications doesn't Implement –nolib_inline
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1
Problem Description	The –nolib_inline option to not inline library function calls does not work in ecc, the compiler for Itanium(R)–based applications.
Resolution/Status	This problem has been resolved in the Intel(R) C++ Compiler 7.1 build l_cc_pu_7.1.013 or later. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http:/www.intel.com/software/products/support.

Title	The exception::what() member function rerturns incorrect result
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	Red Hat* 7.1

Problem Description	The "what" member function of the std::exception class returns incorrect result as shown in the following example:
	$ cat e.cpp
	
	#include <stdio.h>
	namespace std {
	
	class exception {
	public:
	exception () { };
	exception (const exception);
	exception&operator= (const exception);
	virtual ~exception () ;
	virtual const char* what () const ;
	};
	}
	
	int main()
	{
	std::exception e1;
	const char *what = e1.what ();
	printf("what = %s\n",what);
	}
	
	$ icpc exception.cpp
	$ ./a.out
	what = (null) // The correct value should be "what = St9exception"

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_pc_7.1.017 or higher. You may download and install the latest product update from the Premier Support web site at . You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Compiler generates remark #810 for computed if with bitfield
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	SGI Altix ProPack 3.0

Problem Description	When using a computed if conditional expression to evaluate bitfield record member, building with –w2 switch incorrectly reports remark #810. The example below demonstrates the problem.
	
	$ cat test.c
	#include <stdio.h>
	struct BitfieldRec {
	unsigned short a : 1;

	};
	
	int main(int argc, char *argv[])
	{
	unsigned int n;
	struct BitfieldRec r;
	
	n = 123;
	r.a = n ? 0 : 1;
	printf("%u\n",r.a);
	return 0;
	}
	$ icc –w2 test.c
	test.c(12): remark #810: conversion from "int" to "unsigned short" may lose sign
	ificant bits
	r.a = n ? 0 : 1;
	^
	
	$

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, expand the computed if statement
	r.a = n ? 0 : 1;
	to
	if (n)
	r.a = 0;
	else
	r.a = 1;

Title	OpenMP code fails to link if 'write' is used as name for critical region
Product	Intel(R) C++ Compiler for Linux*
Version	7.1
Operating System	Red Hat* 2.1AS

Problem Description	The following code will fail to link.
	$ cat bug.c
	#include <stdio.h>
	#include <omp.h>
	
	main()
	{
	
	#pragma omp parallel
	{
	#pragma omp critical (write)
	{
	printf("Hello\n");
	}

	}
	
	}
	$ ecc –openmp bug.c
	bug.c(8) : (col. 1) remark: OpenMP DEFINED REGION WAS PARALLELIZED.
	ld: /lib/libc.so.6.1: indirect symbol `write' to `write@@GLIBC_2.2' is a loop
	/lib/libc.so.6.1: could not read symbols: Invalid operation

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, use a different name for the critical region.

Title	Compiler reports segmentation violation when compiling C++ files at –ip –ipo
Product	Intel(R) C++ Compiler for Linux*
Version	7.0
Operating System	Red Hat* 7.1

Problem Description	The Itanium compiler reports a segmentation violation error when compiling any C++ file with the "–c –ip –ipo" compiler switches and using the "–lm" switch during the link phase as shown below:
	
	$ ecc –c –restrict –O3 –ip –ipo test.cpp
	$ ecc –o test test.o –ip –ipo –lm
	IPO: using IR for test.o
	IPO: performing single–file optimizations
	ecc: error: Fatal error in /opt/intel/compiler70/ia64/bin/mcpcom, terminated by segmentation violation
	ecc: error: problem during multi–file optimization compilation (code 1)

Resolution/Status	This problem has been resolved in a product update with package ID l_cc_p_8.0.034 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	# and * in asm constraint modifier is not accepted
Product	Intel(R) C++ Compiler for Linux*
Version	8.0
Operating System	SGI Altix ProPack 3.0
Problem Description	If a user's code contains the asm constraint modifiers # and * and compiled with the Intel(R) C++ Compiler for Linux, version 8, an error will occur.

	For example asm volatile ("%0" : : "i#*X"(x)); will produce the following compilation error messages. error: unknown asm constraint modifier '#' asm volatile ("%0" : : "i#*X"(x)); ^ error: unknown asm constraint modifier '*' asm volatile ("%0" : : "i#*X"(x)); This is an issue with the version 7.x of compiler also.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Internal compiler error while compiling CERN*'s ROOT* application version 3.10.2 at –O3 –ipo
Product	Intel(R) C++ Compiler for Linux*
Version	8.0
Operating System	Red Hat Enterprise Linux 3.0
Problem Description	When compiling CERN*'s ROOT* application version 3.10.2 with package ID "root_v3.10.02.source.tar.gz" at –O3 –ipo, the Intel (R) C++ Compiler for Linux 8.0 for Itanium(R) Systems reports the following error while creating the lib/libGUI.so shared library: (0): internal error: 0_0 icc: error: problem during multi–file optimization compilation (code 4)
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	INTEL (flexlm) vendor daemon fails to start
Product	Intel(R) C++ Compiler for Linux*
Version	8.0
Operating System	Red Hat* 9.0
Problem Description	On a system with glibc 2.3 such as Red Hat Linux* 9.0 or Enterprise Linux 3.0, the vendor daemon "INTEL" aborts with the following message: Unknown host: hostname
Resolution/Status	Updated license managers are available on the Intel(R) Premier Support web site at https://premier.intel.com. If you don't have an account, you will need to register your compiler at http://intel.com/software/products/registrationcenter/

	The license managers are automatically available if you have support for the Intel(R) C++ or Fortran Compilers for Linux*, otherwise you will need to request access via Premier Support to the "Intel SW Dev Tools License Servers" product. Alternatively, you can use a different host server for your license manager that runs a supported operating system that is not a flavor of Linux with glibc 2.3. Please see http://support.intel.com/support/performancetools/fortran/license.htm for other operating systems that are supported.

Title	Premature program termination when using shared memory segments greater than 2GB
Product	Intel(R) C++ Compiler for Linux*
Version	8.0
Operating System	Red Hat* Advanced Server 2.1
Problem Description	Using shared memory segments greater than 2GB may cause the executable to terminate with a segmentation fault. This can happen when a shared memory segment greater than 2 GB is created and then any location above the 2 GB limit is addressed for a read/write operation.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Incorrect results produced when code containing complex arithmetic is optimized
Product	Intel(R) C++ Compiler for Linux*
Version	8.0, 8.1
Operating System	Debian 3.1

Problem Description	The function "dummy" performing complex arithmetic on double data types in the example below returns incorrect results when it is compiled at any optimization levels:
	
	#include <complex>
	#include <iostream>
	
	using std::complex;
	using std::cout;
	using std::endl;
	
	complex<double> dummy(complex za) {
	return za + complex<double>(1.0, –1.0);
	}
	
	int main() {
	complex<double> z = complex(0.9283, –0.383);

	cout << z + complex cout << dummy(z) << endl;
	}
	
	$ icc –O2 test.cpp
	$ a.out
	(1.9283,–1.383)
	(1,1.84467e+19) <====== Incorrect value. The correct value should be
	(1.9283,–1.383)

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround compile with "–g" or with the "–cxxlib–icc" switch:
	
	$ icc –O2 –cxxlib–icc test.cpp
	$ a.out
	(1.9283,–1.383)
	(1.9283,–1.383)
	$

Title	__interface variable in some new versions of glibc conflicts with compiler internal
Product	Intel(R) C++ Compiler for Linux*
Version	8.1
Operating System	SUSE* Linux 9.0
Problem Description	When file /usr/include/netinet/in.h is updated to support RFC 3678 "Socket Interface Extensions for Multicast Source Filters", a new argument name in a function prototype called __interface is added. This symbol conflicts with a compiler internal resulting in an error message of "error: invalid combination of type specifiers".
Resolution/Status	This is a known issue that may be resolved in a future product release. The workaround is to add –D__interface=iface to the compiler command line. An alternate workaround is to delete or change __interface in /usr/include/netinet/in.h.

Windows*

Title	Derived class enumeration bug
Reference#	18282
Product	Intel(R) C++ Compiler for Windows*
Version	5.0,6.0,7.0
	Windows* 2000 Server

Operating System	

Problem Description	The construct below results in a compiler error.
	
	class A
	{
	public:
	enum {Enum};
	};
	
	class B : public A
	{
	public:
	enum {Enum=Enum};
	};
	
	error: "Enum" has already been declared in the current scope
	enum {Enum = Enum};
	^
	A workaround is to change one of the enumerated names:
	class A
	{
	public:
	enum {Foo};
	};
	
	class B : public A
	{
	public:
	enum {Enum=Foo};
	};

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	No option to turn off "targeted for automatic cpu dispatch" remarks.
Reference#	19189
Product	Intel(R) C++ Compiler for Windows*
Version	5.0,6.0,7.0
Operating System	Windows NT* 4.0 Service Pack 4

Problem Description	With foo.cpp:
	---- begin foo.cpp ----
	void addOne(unsigned char *pData, unsigned int length)
	{
	for(unsigned int i = 0; i < length; i++)
	{
	pData[i] += 1;

	} } ---- end foo.cpp ---- ..."icl /c /QaxM foo.cpp" gives: foo.cpp foo.cpp(3) : (col. 5) remark: LOOP WAS VECTORIZED. foo.cpp(2) : (col. 1) remark: ?addOne@@YAXPAEI@Z has been targeted for automatic cpu dispatch. There is no option to turn off the remarks about functions being targeted for automatic cpu dispatch.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	printf invalid format string: warning #269
Reference#	20109
Product	Intel(R) C++ Compiler for Windows*
Version	5.0.1, 6.0,7.0
Operating System	Windows* 2000 Server

Problem Description	There are cases where the compiler reports invalid format string conversion in printf/fprintf/etc. functions when they are valid. This happens when using "%S" to specify wide character strings when using the C runtime library's printf (also "%ws", "%wS", "%ls" and "%lS") or to specify single character strings when using the C runtime library's wprintf (also "%hs" and "%hS").
	
	#include <stdio.h>
	
	int main(int argc, char *argv[])
	{
	wchar_t widestring[] = L"A Wide String";
	char narrowstring[] = "A Narrow String";
	
	char widedest[100];
	char narrowdest[100];
	
	// Convert a wide string to a narrow string – 5 different ways
	sprintf(narrowdest, "%S", widestring); // because we are a non–wide char app, "%S" means a wide string.
	// If we were a wide char app, "%S" would mean a normal char string
	sprintf(narrowdest, "%ls", widestring);
	sprintf(narrowdest, "%lS", widestring);
	sprintf(narrowdest, "%ws", widestring);
	sprintf(narrowdest, "%wS", widestring);

	// Convert a narrow string to a wide string – 2 different ways
	sprintf(widedest, "%hs", narrowstring);
	sprintf(widedest, "%hS", narrowstring);
	
	return 0;
	}

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Itanium(R) compiler unaligned access error with __unaligned keyword used on a structure
Reference#	21906
Product	Intel(R) C++ Compiler for Windows*
Version	5.0,6.0.7.0
Operating System	Windows XP
Problem Description	The Itanium(R) comipiler doesn't create the proper unaligned stores when trying to store unaligned data from a structure. This causes an unaligned access error when the compiled program is run.
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, try one of the following: 1) copy each member of the struct separately (so the operator= does not come into play.) 2) Define a user defined operator= that takes an unaligned object. For example: __unaligned Object&operator=(const Object)__unaligned {}

Title	Compiler does not support double casting
Reference#	22666
Product	Intel(R) C++ Compiler for Windows*
Version	6.0,7.0
Operating System	Windows* 2000 Professional
Problem Description	The Intel(R) C++ compiler for Itanium(R)–based applications does not support double casting. When you try to double cast with the Intel compiler, the compiler gives an "integer conversion resulted in truncation" error. Here is a code example that reports this error: const unsigned __int32 addr32 = (unsigned __int32) (unsigned __int64) // Note

	the double cast (first to __int64 then to __int32)
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Dependency information not generated under certain circumstances
Reference#	22910
Product	Intel(R) C++ Compiler for Windows*
Version	5.0.1, 6.0, 7.0
Operating System	Windows* 2000 Professional
Problem Description	If a header file is updated in an application, pressing F7 in Microsoft* Visual Studio 6 does not rebuild the application. The problem is that initial dependency information is not created because the Microsoft compiler finds an error in the header file but the Intel compiler does not. Updating the header file further will also not be recognized since there is no dependency information to reference.
Resolution/Status	This problem has been resolved in a product update with package ID W_CCX_P_8.0.040 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support.

Title	Large value (>=0x80000000) cast from double to int produces incorrect results
Reference#	23572
Product	Intel(R) C++ Compiler for Windows*
Version	6.0,7.0
Operating System	Windows* 2000 Professional

Problem Description	A large double precision value that is cast from double to int produces incorrect results.
	
	For example, the following code fragment:
	
	double v=2164195328.0000; // (0x80ff0000)
	int optval = (int)(v+0.5);
	
	Results in optval=0x80000000

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround first cast to unsigned int and then to int.
	
	Example:
	
	double v=2164195328.0000; // (0x80ff0000)
	int optval;
	
	if (v optval = –(int)(unsigned int)(–v+0.5);
	else
	optval = (int)(unsigned int)(v+0.5);

Title	Apps with Deep Class Inheritance such as STLport will take Longer to Compile
Reference#	25496
Product	Intel(R) C++ Compiler for Windows*
Version	7.0, 7.1
Operating System	Windows* 2000 Professional
Problem Description	We have determined that the Intel(R) C++ Compiler has some compile time issues related to the depth of class inheritance. These issues cause an application with deep class inheritance to require excessive compile time..
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	The C++ Compiler compiles an illegal member–like constructor call
Reference#	24227
Product	Intel(R) C++ Compiler for Windows*
Version	
Operating System	Windows* 2000 Professional

Problem Description	ICL compiles successfully the code where a class constructor is called within the body of different consructor of the same class.
	Other compilers (VC6, gcc) mark this code as error.
	
	class A {
	public:
	A();
	A(int);
	int i_;
	};
	
	A::A(int i) {
	i_ = i;

	}
	
	A::A() {
	this–>A(2);
	}

Resolution/Status	This problem has been resolved in a product update with package ID W_CC_PU_7.0.073 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Compiler issues error message when compiling in–line asm code using __asm_emit
Reference#	25485
Product	Intel(R) C++ Compiler for Windows*
Version	6.0, 7.0
Operating System	Windows* 2000 Professional

Problem Description	The Intel C++ Compiler for Windows (icl), issues an error when compiling the following code:
	
	#define CPUID_INSTRUCTION \
	__asm _emit 0x0f \
	__asm _emit 0xa2
	#endif
	
	int __glQueryCPUID()
	{
	__asm CPUID_INSTRUCTION
	return 0;
	}
	
	error: label "_emit" was referenced but not defined

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, do not specify the __asm keyword on "_emit 0x0f" as shown below: #define CPUID_INSTRUCTION \ _emit 0x0f \ __asm _emit 0xa2 #endif int __glQueryCPUID()

	{
	__asm CPUID_INSTRUCTION
	return 0;
	}

Title	Itanium(R) compiler treats the labs() intrinsic function like a regular function call
Reference#	25649
Product	Intel(R) C++ Compiler for Windows*
Version	7.0, 7.1
Operating System	Windows* 2000 Professional

Problem Description	Despite declaring the labs() intrinsic function as specified in the compiler documentation, the Itanium(R) compiler still emits a library call when the caller of labs() uses the return value of labs() as shown below:
	
	long labs(long);
	
	long ltest(long l) {
	return labs(l);
	}
	ecl –c –Fa test.c
	more test.asm|grep labs
	br.call.sptk.many b0=labs# ;; //1: 4 5. Compiler has treated labs as a function call rather than as an intrinsic
	.type labs#,@function
	.global labs#

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Runtime errors when using Stingray* Objective Studio 2000 with the C++ Compiler for Windows*
Reference#	26318
Product	Intel(R) C++ Compiler for Windows*
Version	6.0,7.0
Operating System	Windows* 2000 Server
Problem Description	Run–time errors such a invalid memory accesses may result when using the Intel(R) C++ Compiler for Windows* to compile an application that makes use of

	the Stingray* Objective Studio 2000 class library.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	icl ignores the Using−declaration of base class assignment operator
Reference#	28272
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* 2000 Professional
Problem Description	icl ignores the using−declaration of the base class assignment operator. As a result, the following code code will not compile: struct A { A&operator =(const A) }; struct B : A { B&operator =(const B) using A::operator =; }; int main() { B b; A a; b = a; } t.cpp t.cpp(2): warning #784: using−declaration of function "A::operator=" ignored struct B : A { B&operator =(const B) using A::operator =; }; ^ t.cpp(3): error: no operator "=" matches these operands operand types are: B = A int main() { B b; A a; b = a; }
Resolution/Status	This problem has been resolved in the Intel C++ Compiler version 8.0. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit

Title	Pragma warning directive turns all warnings to errors
Reference#	29485
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* 2000 Professional

Problem Description	The pragma warning directive of the form "#pragma warning (error: xxxx)", where xxxx is the warning number, causes the compiler to report all warnings as errors rather than reporting only warning number xxxx as an error.
Resolution/Status	This problem has been resolved in a product update with package ID W_CC_PC_8.0.28 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Definition for explicit instantiations of template functions is required
Reference#	30900
Product	Intel(R) C++ Compiler for Windows*
Version	6.0,7.0
Operating System	Windows* 2000 Professional
Problem Description	Compiling code with template functions with a declaration of an explicit instantiation of a template function, but no function definition of the explicit instantiation, causes a compiler error: "error: function "foo(T *, int, T) [with T=double]" cannot be instantiated –– no template definition was supplied template int foo<double>(double *x, int y, double z);" This is not an error in the Microsoft* Visual C++* 6.0 compilers or above.
Resolution/Status	This problem has been resolved in a product update with package ID W_CC_PU_7.0.094 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	No warning message issued when 'delete' is called on a local array
Reference#	29945
Product	Intel(R) C++ Compiler for Windows*
Version	6.0, 7.0
Operating System	Windows* 2000 Professional
Problem Description	In the following code the 'delete' operator is called on a local array that has not been created dynamically. Therefore, the compiler should issue a warning message, but it doesn't.

	main()
	{
	char test[10];
	delete [] test; // Compiler should issue warning for this statement because the array 'test' is not dynamic (e.g. was not created with the 'new' operator).
	};
	
	icl –c test.cpp
	Intel(R) C++ Compiler for 32–bit applications, Version 6.0.1 Build 20021015Z
	Copyright (C) 1985–2002 Intel Corporation. All rights reserved.
	
	test.cpp // Code compiled with no warning !

Resolution/Status	This is a known issue that may be resolved in a future product release. In the mean time, please ensure that you do not use "delete" on an array as shown in the above example.

Title	Preprocessing goes into endless loop when the source includes itself
Reference#	41075
Product	Intel(R) C++ Compiler for Windows*
Version	
Operating System	Windows* 2000 Professional

Problem Description	When a source file includes its own pre–processed file, compilation with /E or /P goes into an infinite loop. In the following example the file name "foo.cpp" includes the pre–processed file with the same name "foo.i"
	
	foo.cpp:
	
	#include <iostream.h>
	#include "foo.i"
	int main(int argc, char **argv)
	{
	cout << "Hello world!" << endl;
	return 0;
	}

Resolution/Status	This problem has been resolved. The solution will be available in a future product release.

Title	Reference to zero–sized array not reported as error
Reference#	31326

Product	Intel(R) C++ Compiler for Windows*
Version	7.1
Operating System	Windows* 2000 Professional

Problem Description	The following code should report an error on line 1 due to referencing a zero−sized array, but it does not.
	
	void f(int (
	{
	}

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Additional parentheses in sizeof() argument causes the Intel(R) C++ Compiler to diagnose an error
Reference#	31487
Product	Intel(R) C++ Compiler for Windows*
Version	6.0,7.0
Operating System	Windows* XP Professional

Problem Description	Consider the following code.
	
	struct B { B() {} };
	struct A { A(B){} A(int) {} };
	
	const int c = sizeof A(B()); // Example 1
	const int d = sizeof A((B())); // Example 2
	
	The Intel(R) C++ Compiler compiles Example 1 without error, but on Example 2, an error is flagged: "error: operand of sizeof may not be a function". This causes a compatibility issue with Microsoft* Visual C++* which compiles both lines without problems.
	
	The problem is due to the fact that both lines violate the ANSI C++ Standard (see section 8.2 of the ANSI standard for details). The Intel compiler compiled Example 1 anyway so as to maintain Microsoft compatibility, but the Intel compiler did not recognize that Example 2 was equivalent to Example 1 and diagnosed an error.

Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	extern "C" comment() definition compiled with –Zi causes an internal compiler error
Reference#	28538
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional
Problem Description	When code like the following is compiled with –Zi, an internal compiler error will be emitted. extern "C" void comment() {} The internal compiler error is due to the combination of the extern "C", the function definition being named "comment" (or any section name such as text, data, bss, sdata etc), and generating debug information using –Zi. Without any one of these factors, this code will compile normally.
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, rename the function to a name that is not the name of a section.

Title	Unused static functions are not removed and may cause link errors
Reference#	21753
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional

Problem Description	If a static function is defined in a program but never called, the compiler does not eliminate the code. If one of these unused functions calls another function that is not defined, there will be a link time error as shown below:
	––––––––––––––––
	extern void foo();
	
	static void nttwait()
	{
	foo();
	}
	
	void main()
	{
	if (0) nttwait();
	}
	––––––––––––––––
	
	– Compile the test case as "icl –c –O2 test.c"
	– Use "dumpbin /symbol test.obj". Notice that the generated object file contains

	a reference to the static function nttwait() that's never called in the program. If the function foo() is not defined anywhere, error will occur at link time as shown below: icl test.c –out:test.exe test.obj test.obj : error LNK2019: unresolved external symbol _foo referenced in function _nttwait test.exe : fatal error LNK1120: 1 unresolved externals
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Aligned and unaligned intrinsics not differentiated when inlined
Reference#	33117
Product	Intel(R) C++ Compiler for Windows*
Version	7.0,7.1
Operating System	Windows* XP Professional
Problem Description	If inlined, a function that determines if data is aligned and then calls the appropriate aligned or unaligned intrinsic might be converted into a simple call to the unaligned version of the intrinsic regardless of the data alignment. The optimizer does not correctly recognize the aligned/unaligned nature of these intrinsics and treats them as the same intrinsic. This causes the optimizer to optimize away the alignment check and the branching as well and merge the two calls into one – the unaligned intrinsic. An example of such intrinsics would be the _mm_loadu_si128 intrinsic (unaligned) and the _mm_load_si128 intrinsic (aligned).
Resolution/Status	This problem has been resolved in the Intel(R) C++ Compiler 7.1 W_CC_PC_7.1.014 or later. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http:/www.intel.com/software/products/support.

Title	The _OPENMP predefined macro is not documented
Reference#	32992
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional

Problem Description	The _OPENMP predefined macro is not documented in the User's Guide in the predefined macros section. The _OPENMP macro is defined by the compiler as described in the OpenMP C and C++ Application Program Interface Version 2.0 found at http://www.openmp.org/specs/.
Resolution/Status	This problem has been resolved in a product update with package ID W_CCX_P_8.0.040 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support.

Title	PGO Performance Counter Overflow Causes Erroneous "Low Trip Count" Diagnostics
Reference#	33423
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Advanced Server
Problem Description	If the execution counts in a loop are particularly high, they can overflow the compiler's execution counters during the Profile Generation phase of the Profile−Guided Optimizations (/Qprof_gen). This causes incorrect, low execution counts which can affect many optimizations in the compiler, especially those that check loop trip counts. One key optimization that is affected is Software Pipelining which will not occur on loops with low trip counts. If you see "often−used" loops designated as having low trip counts in the OpenMP*, Vectorization, or Optimization reports when you are using /Qprof_use, counter overflow may be the cause.
Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, the compiler sometimes supplies directives to override the compiler's decision to not optimize loops (for example, "#pragma vector always" and "#pragma swp").

Title	7.1 IDE Integration Overwrites 6.0 Integration
Reference#	33931
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional
Problem Description	When installing the Intel(R) C++ Compiler 7.x and integrating with Microsoft Visual C++* .NET 2002, if you have the Intel(R) C++ Compiler 6.0 already integrated, the installation will ask whether you want to reinstall the integration

	utility. Regardless of whether you choose to reinstall or not, the 7.x integration will overwrite the 6.0 integration. You can see this by going to Tools–>Options. Now there will be an "Intel C++" entry with the 7.x settings. The 6.0 "Intel Configuration Options" entry is gone.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Pure Virtual Functions Cannot Be Set With Integer Specifiers
Reference#	34091
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional

Problem Description	The following code will not compile with the Intel compiler:
	
	class x {
	virtual void foo() = 0L;
	};
	
	You will get an error:
	
	error: badly–formed pure specifier (only "= 0" is allowed)
	
	Microsoft* Visual Studio* .NET accepts this syntax.

Resolution/Status	This problem has been resolved in the Intel(R) C++ Compiler for Windows* 8.0 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com/. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/. As a workaround, remove the 'L' integer literal specifier.

Title	dllexport Attribute not Correctly Applied to Explicit Template Specializations
Reference#	34271
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* 2000 Server
Problem Description	Currently if a function in a template class is specialized, the Intel(R) C++ Compiler ignores any declspec(dllexport) attribute. That isn't what Microsoft* Visual C++* does though. If it is a function template, Microsoft uses the declspec specified on the specialization which includes removing dllexport if it was on the original function but not on the specialization. For other template

	specializations it looks like a merge of the dllexport attribute, so if the attribute is on the original or on the specialization, then the specialization is dllexport.
Resolution/Status	This problem has been resolved in the Intel(R) C++ Compiler for Windows* 8.0 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com/. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/.

Title	Compile error when both a template copy constructor and a template conversion "operator =" defined
Reference#	16886
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional
Problem Description	For classes that have both a template copy constructor and a template conversion "operator =" explicitly defined, the Intel compiler will give an error when a conversion from one class instantiation to another is needed. For example: Car<fivespeed> myCar; Car<hotrod> myHotRod = myCar; would get the error: error: more than one user–defined conversion from "Car<fivespeed> " to "Car<hotrod>" applies: function template "Car<T>::operator Car() [with T=fivespeed]" function template "Car<T>::Car(Car)[with T=hotrod]" Car<hotrod> myHotRod = myCar; ^
Resolution/Status	This problem has been resolved in a product update with package ID W_CC_PC_8.0.041 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com/. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/.

Title	std::allocator::rebind compilation error
Reference#	34653

Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* 2000 Professional
Problem Description	The compiler issues an error for a template with following code: typedef allocator_type::rebind <value_type>::other value_alloc_type; where allocator_type is "std::allocator<std::pair" The error is: class "std::allocator<std::pair" has no member "rebind" typedef allocator_type::rebind <value_type>::other value_alloc_type;
Resolution/Status	This is not a defect of the Intel(r) compiler. The Intel compiler uses the header files that Microsoft* ships with its Visual C++* compiler. The problem occurs because the allocator class that is declared in the Visual C++* 6.0 version of does not contain a rebind type. The Microsoft Visual C++* .NET 2002 or .NET 2003 versions of do declare this type. If you have one of the latter versions of Visual C++ installed and integrated with the Intel compiler, you will not receive this compilation error.

Title	Unresolved Symbols with –MD[d] and Microsoft* Visual C++* .NET 2003
Reference#	33872
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional

Problem Description	Code like the following:
	
	template <class T>
	class Basic {
	public:
	virtual ~Basic() {}
	void foo(int);
	void goo();
	};
	
	template <>
	void Basic<int>::foo(int) {}
	
	template class __declspec(dllimport) Basic<int>;
	
	will cause the linker to emit unresolved external symbols when you build the objects with the Intel(R) C++ Compiler integrated with Microsoft* Visual C++* .NET 2003 and you use /MD or /MDd. These errors will look like:
	
	hello.obj : error LNK2019: unresolved external symbol "const std::basic_ist

	ream<unsigned short,struct std::char_traits::`vftable'" (??_7? $basic_istream@GU?$char_traits@G@std@@@std@@6B@) referenced in function "public: virtual __thiscall std::basic_istream<unsigned short,struct std::char_traitssigned short> >::~basic_istream<unsigned short,struct std::char_traitsshort> >(void)" (??1?$basic_istream@GU?$char_traits@G@std@@@std@@UAE@XZ) hello.obj : error LNK2019: unresolved external symbol "const std::basic_ist ream<unsigned short,struct std::char_traits::`vbtable'" (??_8? $basic_istream@GU?$char_traits@G@std@@@std@@7B@) referenced in function "public: __thiscall std::basic_istream<unsigned short,struct std::char_traitshort> >::basic_istream<unsigned short,struct std::char_traits >(enum std::_Uninitialized)" (??0?$basic_istream@GU?$char_traits@G@std@@@std@@QAE@ W4_Uninitialized@1@@Z) hello.obj : error LNK2019: unresolved external symbol "const std::basic_ost ream<unsigned short,struct std::char_traits::`vftable'" (??_7? $basic_ostream@GU?$char_traits@G@std@@@std@@6B@) referenced in function "public: virtual __thiscall std::basic_ostream<unsigned short,struct std::char_traitssigned short> >::~basic_ostream<unsigned short,struct std::char_traitsshort> >(void)" (??1?$basic_ostream@GU?$char_traits@G@std@@@std@@UAE@XZ) hello.obj : error LNK2019: unresolved external symbol "const std::basic_ost ream<
Resolution/Status	This problem has been resolved in a product update with package ID W_CC_PC_7.1.016 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com/. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/. As a workaround, use another of the Microsoft Runtime Library options (such as /ML or /MT) if possible.

Title	Intel Specific property does not show under Microsoft* Visual Studio .NET 2002 Japanese Edition
Reference#	34674
Product	Intel(R) C++ Compiler for Windows*
Version	7.1
Operating System	Windows* 2000 Professional
Problem Description	The "Intel Specific" property in the Intel Compiler Integration Tool does not show under the Japanese Language Edition of the Microsoft* Visual Studio .NET 2002. The problem exists even after the Intel Compiler Intergration Tool has been enabled.

Resolution/Status	This problem has been resolved in a product update with package ID w_cc_p_8.0.028 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	/GL (Whole Program Optimization) switch may increase object size
Reference#	36069
Product	Intel(R) C++ Compiler for Windows*
Version	7.1
Operating System	Windows* XP Professional
Problem Description	Compiling a C++ program using the /GL switch may increase the size of the generated object file by 10 fold.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Internal Compiler Error on Code with Multiple goto Statements
Reference#	37449
Product	Intel(R) C++ Compiler for Windows*
Version	7.0
Operating System	Windows* XP Professional

Problem Description	The Intel(R) C++ Compiler for Windows* may report internal compiler error when compiling code that contains multiple "goto" statements like the example shown below. The error will occur regardless of optimization level, but has only been exhibited on the compiler for IA32–based applications.
	
	void test()
	{
	int k;
	if (k <= 100)
	{
	L1:
	if (k >= 50)
	if (k <= 90)
	goto L2;
	else
	goto L2;

	goto L1; } L2: return; }
Resolution/Status	This problem has been resolved in a product update with package ID w_cc_pc_7.1.024 or higher. You may download and install the latest product update from the premier support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Unresolved symbol error "__VEC_memzero" when calling the memset function
Reference#	37176
Product	Intel(R) C++ Compiler for Windows*
Version	7.1, 8.0
Operating System	Windows* XP Professional

Problem Description	Unresolved symbol error "__VEC_memzero" is reported when compiling code that uses memset function as shown below. Calling other memory functions may cause other unresolved errors. The errors occur if the Intel Compiler Libraries libirc or libircmt are not linked to the application:
	
	>type test.c
	main()
	{
	void *ptr;
	memset(ptr, 0, 0x8000L);
	}
	
	>icl test.c /link /nodefaultlib
	Intel(R) C++ Compiler for 32−bit applications, Version 7.1 Build 20030924Z
	Copyright (C) 1985−2003 Intel Corporation. All rights reserved.
	
	test.c
	Microsoft (R) Incremental Linker Version 7.10.3077
	Copyright (C) Microsoft Corporation. All rights reserved.
	
	−out:test.exe
	test.obj
	test.obj : error LNK2019: unresolved external symbol __VEC_memzero referenced in
	function _main

	LINK : error LNK2001: unresolved external symbol _mainCRTStartup test.exe : fatal error LNK1120: 2 unresolved externals
Resolution/Status	This is a known issue that may be resolved in a future product release. The current solution is to link with the Intel Compiler Libraries libirc or libircmt.

Title	The FFTW library does not compile when using the –Qipo switch
Reference#	38477
Product	Intel(R) C++ Compiler for Windows*
Version	8.0
Operating System	Windows* 2000 Professional
Problem Description	The Intel(R) C++ Compiler does not compile the FFTW (http://www.fftw.org) library if the /Qipo switch is used.. An error is generated in the file dft–vrank–geq1.c. The error is "(0): internal error: backend signals." The FFTW DLL correctly compiles and links without the /Qipo switch.
Resolution/Status	This problem has been resolved in a product update with package ID w_cc_pc_8.0.048 or higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support

Title	Problem with project rebuilding when ici80.dg is deleted
Reference#	38225
Product	Intel(R) C++ Compiler for Windows*
Version	8.0
Operating System	Windows* XP Professional
Problem Description	The ici80.dg file is located in the debug or release directory of the project. If the ici80.dg file is deleted, this results in the rebuilding of sources that have Custom Build Steps. This only applies to the files with Custom Build Steps.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	The _mm_malloc() intrinsic function doesn't return zero on failure
Reference#	38704
Product	Intel(R) C++ Compiler for Windows*

Version	7.1
Operating System	Windows* XP Professional
Problem Description	The _mm_malloc() intrinsic function throws an exception instead of returning zero when insufficient memory is available.
Resolution/Status	This issue has been resolved in the compiler package w_cc_pc_8.0.048 or higher.

Title	The Itanium(R) Assembler (ias) does not flag error for unbalanced curly or square brackets
Reference#	41535
Product	Intel(R) C++ Compiler for Windows*
Version	IAS 7.0
Operating System	Windows* XP Professional
Problem Description	Itanium assembler accepts instructions with unbalanced square brackets without flagging error. It accepts code like: ld8 r2=[r3,8 st8 [r2=r3,8 and treats it as if it had been: ld8 r2=[r3],8 st8 [r2]=r3,8
Resolution/Status	This is a known issue which may be resolved in a future product release

Title	Unresolved external symbol ___readfsdword referenced
Reference#	39854
Product	Intel(R) C++ Compiler for Windows*
Version	8.0
Operating System	Windows* XP Professional
Problem Description	The Intel(R) C++ Compiler for Windows* does not recognize the "readfsdword" intrinsic function which is invoked when using the GetCurrentFiber() function as shown below: #include <windows.h> #include <stdio.h> void main() { LPVOID t = GetCurrentFiber();

	printf("Result = %p\n", t);
	}
	
	>>icl test.c
	Intel(R) C++ Compiler for 32−bit applications, Version 8.0 Build 20040415Z P
	kage ID: W_CC_PC_8.0.048_PE049.1
	Copyright (C) 1985−2004 Intel Corporation. All rights reserved.
	
	test.c
	Microsoft (R) Incremental Linker Version 7.10.3077
	Copyright (C) Microsoft Corporation. All rights reserved.
	
	−out:test.exe
	test.obj
	test.obj : error LNK2019: unresolved external symbol ___readfsdword
	referenced
	n function _main
	test.exe : fatal error LNK1120: 1 unresolved externals

Resolution/Status	This is a known issue that may be resolved in a future product release. As a workaround, you can put the definition of the readfsdword instrinsic into a seaprate C source file and link with the main program as shown below:
	
	>>type readfsdword.c
	#pragma warning (disable:4035) // disable 4035 (disable the function must return a value warning)
	void * __readfsdword(void)
	{
	__asm mov eax, fs:[0x10]
	}
	#pragma warning (default:4035) // Reenable the warning
	
	>>icl −c readfsdword.c
	Intel(R) C++ Compiler for 32−bit applications, Version 8.0 Build 20040519Z Pac
	kage ID: W_CC_PC_8.0.048_PE050.1
	Copyright (C) 1985−2004 Intel Corporation. All rights reserved.
	
	readfsdword.c
	
	>>icl test.c readfsdword.obj
	Intel(R) C++ Compiler for 32−bit applications, Version 8.0 Build 20040519Z Pac
	kage ID: W_CC_PC_8.0.048_PE050.1
	Copyright (C) 1985−2004 Intel Corporation. All rights reserved.
	
	test.c
	Microsoft (R) Incremental Linker Version 7.10.3077

	Copyright (C) Microsoft Corporation. All rights reserved. –out:test.exe test.obj readfsdword.obj >>test.exe Result = 00001E00

Title	Abstract virtual function called from constructor not diagnosed until runtime
Reference#	45537
Product	Intel(R) C++ Compiler for Windows*
Version	8.1
Operating System	Windows* XP Professional

Problem Description	Intel(R) C++ Compiler for Windows should NOT compile the following sample code:
	
	#include <stdio.h>
	class base
	{
	public:
	virtual void printme() = 0;
	};
	
	class derived1 : public base
	{
	public:
	derived1() { printme(); }
	};
	
	class derived2 : public derived1
	{
	public:
	void printme() { printf("printme()\n"); }
	};
	
	void main()
	{
	derived2 d;
	d.printme();
	}
	
	Explanation:
	During the execution of the constructor for class derived1, the dynamic type of the object being construct is derived1, even at some later point in time, the type

	of the object will become derived2. And a virtual function of class derived2 can not be called for an object of class derived1. So when the printme function is invoked, derived2::printme can't be called, derived2 has no printme function. That leaves only base::printme, which is pure virtual– there is no function to call. This should be caught at compile–time.
Resolution/Status	This is a known issue that may be resolved in a future product release.

Title	Name–mangling of __stdcall Constructors/Destructors with Mixed Builds
Reference#	45977
Product	Intel(R) C++ Compiler for Windows*
Version	8.0
Operating System	Windows* 2000 Professional

Problem Description	If you declare a class constructor or destructor with a __stdcall attribute, the Microsoft Visual C++* compiler (correctly) ignores this attribute and emits the __thiscall'ed mangled name. The Intel(R) C++ Compiler for Windows* emits the __stdcall'ed name instead. The following example illustrates the problem:
	
	class Foo {
	public:
	#ifdef BUILD_DLL
	__declspec(dllexport) __stdcall Foo::Foo(unsigned int);
	#elif USE_STDCALL
	__declspec(dllimport) __stdcall Foo::Foo(unsigned int);
	#else // work around
	__declspec(dllimport) Foo::Foo(unsigned int);
	#endif
	private:
	unsigned int m_x;
	};
	
	#ifdef BUILD_DLL
	Foo::Foo(unsigned int x) {
	m_x = x;
	}
	#else
	int main(void) {
	unsigned int y = 3;
	Foo x(y);
	return(0);
	}
	#endif
	
	// To see the linker error perform the following steps: 1. compile the dll with cl; 2. link the dll with icl.

	>cl –Od –D BUILD_DLL test.cpp –link –dll >icl –Od –D USE_STDCALL test.cpp –link test.lib and you will get the error: test.obj : error LNK2019: unresolved external symbol "__declspec(dllimport) public: __stdcall Foo::Foo(unsigned int)" (__imp_??0Foo@@QAG@I@Z) referenced in function _main test.exe : fatal error LNK1120: 1 unresolved externals // The workaround is to remove "__stdcall" when importing. The following command works for this example: >icl –Od test.cpp –link test.lib
Resolution/Status	This problem has been resolved in versions 8.1 and higher. You may download and install the latest product update from the Premier Support web site at https://premier.intel.com/. You need to be a registered user to access Premier Support. For registration information, please visit http://www.intel.com/software/products/support/.

Title	Error LNK2001 when compiling projects created by ATL Wizard
Reference#	48977
Product	Intel(R) C++ Compiler for Windows*
Version	8.0
Operating System	Windows* XP Professional
Problem Description	Error LNK2001 is produced when the Intel(R) C++ Compiler for IA–32 Applications compiles a default ATL project created by the Microsoft* .NET 2003 ATL Wizard as shown in the following example: – Click File\New\Project – Click Visual C++ Project\ATL\ATL Project – Uncheck the "Attributed" checkbox under the "Application Settings" – Switch to 'Class View' – Right–click on the projectname and select 'Add Class' – Add an ATL Simple Object – Convert to use Intel C++ Project System – Build the project You'll get the following unresolved errors during the link phase: atlbugnoattribute.obj : error LNK2001: unresolved external symbol _LIBID_atlbugnoattributeLib SimpleATLObject.obj : error LNK2001: unresolved external symbol _LIBID_atlbugnoattributeLib SimpleATLObject.obj : error LNK2001: unresolved external symbol

	_IID_ISimpleATLObject Debug/atlbugnoattribute.dll : fatal error LNK1120: 2 unresolved externals
Resolution/Status	This is a known issue that may be resolved in a future product update.

Title	Missing copy constructor instantiation at –Od when the class contains "__declspec(dllimport)"
Reference#	49660
Product	Intel(R) C++ Compiler for Windows*
Version	
Operating System	Windows* XP Professional

Problem Description	Building the following sample code with the Intel(R) C++ Compiler for IA32 Applications at the /Od optimization level results in an unresolved symbol error during the link phase as shown below:
	
	struct __declspec(dllimport) A {
	A() {}
	int i[20];
	};
	
	struct B : public A {
	};
	
	void foo(B b)
	{
	}
	
	int main()
	{
	B b;
	foo(b);
	return 0;
	}
	
	icl –Od test.cpp
	Intel(R) C++ Compiler for 32–bit applications, Version 8.1 Build 20041119Z
	Package ID: W_CC_PC_8.1.022
	Copyright (C) 1985–2004 Intel Corporation. All rights reserved.
	
	test.cpp
	Microsoft (R) Incremental Linker Version 7.10.3077
	Copyright (C) Microsoft Corporation. All rights reserved.
	
	–out:e.exe

	test.obj test.obj : error LNK2019: unresolved external symbol "__declspec(dllimport) public: __thiscall A::A(void)" (__imp_??0A@@QAE@XZ) referenced in function "public: __t hiscall B::B(void)" (??0B@@QAE@XZ) test.exe : fatal error LNK1120: 1 unresolved externals
Resolution/Status	This is a known issue that may be resolved in a future product release. This problem only occurs at the /Od optimization level. As a workaround, compile the code at other optimization levels such as /O1, /O2, etc.

Title	Wrong vectorization if "#pragma novector" is not used in test case
Reference#	49807
Product	Intel(R) C++ Compiler for Windows*
Version	8.1
Operating System	Windows* 2000 Professional

Problem Description	The following program gives an incorrect result if "#pragma novector" (as seen on line 17) is not used before a loop.
	
	1. #include <stdio.h>
	2. #include <math.h>
	3. #include <malloc.h>
	
	4. #define LARGE_EXPONENTIAL_THRESH 60
	
	5. static inline double mydpsi(double x,double nu) {
	6. double tmp=(1.0/nu)*x;
	7. double tmp2=fabs(tmp);
	8. if (tmp2>LARGE_EXPONENTIAL_THRESH) return tmp/tmp2;
	9. else {
	10. double e=exp(2*tmp);
	11. return (e−1)/(e+1);
	12. }
	13. }
	
	14. void test_dpsi(int dim,double* dw,double* w0,double* w,double nu,double l1_lambda)
	15. {
	16. int j;
	17. #pragma novector
	18. for (j=0;j<dim;j++) {

```
19. dw[j]+=l1_lambda*mydpsi(w0[j]+w[j],nu);
20. }
21. }

22. void test_test_dpsi()
23. {
24. int i;
25. int dim=1000;
26. double* dw=new double[dim];
27. double* w0=new double[dim];
28. double* w=new double[dim];
29. double nu=0.0390625;
30. double l1_lambda=1.0;
31. for (i=0;i<dim;i++) {
32. w0[i]=3.2;
33. w[i]=0.0654883;
34. dw[i]=0.0;
35. }
36. test_dpsi(dim,dw,w0,w,nu,l1_lambda);
37. for (i=0;i<100;i++) {
38. printf("%-4d %-16lg\n",i,dw[i]);
39. }
40. delete [] dw;
41. delete [] w0;
42. delete [] w;
43. }

44. void main() {

45. test_test_dpsi();
46. }
```

Resolution/Status	This is a known issue that may be resolved in a future product release.